REMARKS

This Amendment is filed in response to the Office action mailed October 19, 2010. All objections and rejections are respectfully traversed.

Claims 1 – 15, 19 – 20, and 23 – 42 are pending in this case.

Claims 1, 9, 19 – 20, 23, 31, and 39 – 42 were amended.

Interview Summary

On December 8, 2010 and January 5, 2011 the Applicant's attorney conducted telephone interviews with the Examiner. The Applicant thanks the Examiner for his time. Representative claim 1, claim 2, and cited references Federwisch et al., U.S. Patent Publication No. 2003/0182313 (hereinafter "Federwisch"), Edwards, U.S. Publication No. 2003/0182389 (hereinafter "Edwards"), and Haskin et al., U.S. Publication No. 2003/0158863 (hereinafter "Haskin") were discussed. The Applicant drew the Examiner's attention to the limitation " … ***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store***" and "***filling each hole in the writeable vdisk by replacing each invalid pointer with a pointer to the data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created***" of claim 1. Further, the Applicant drew the Examiner's attention to the limitation "***dirtying the data block pointed to by the backing store indirect block to enable write allocation of the dirty data block without altering a data content of the data block***" of claim 2. The Examiner stated that the claims were novel over the prior art of record and that the Examiner would contact the Applicant's attorney before issuing the next Office Action.

Declarations

At pages 20 – 22 of the Office Action, the Examiner declined to accept the supplement declarations filed on August 4, 2010 under 37 CFR 1.131. Rather than obtain further declarations, the Applicant traverses the rejections as discussed during the telephone interviews.

Claim Rejection – 35 USC §103

At page 2 of the Office Action claims 1, 3, 6 – 10, 13 – 15, 19, 20, 23, 25, 28 – 31, 33, and 36 – 42 were rejected under 35 U.S.C §103(a) as being unpatentable over Federwisch in view of Edwards.

The present invention, as set forth in claim 1, representative of claims 1, 3, 6 – 10, 13 – 15, 19, 20, 23, 25, 28 – 31, 33, and 36 – 39 comprises:

1. A method for operating a data storage system, comprising:

creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created;

maintaining a backing store, the backing store referencing data stored in the data storage system which has not been changed since the writable vdisk was created;

loading blocks of the writable vdisk into a memory, the loaded blocks including a writable vdisk indirect block having a plurality of fields, each field storing a valid pointer to a data block or an invalid pointer representing one of a plurality of holes, ***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store***;

loading blocks of the backing store into the memory, the loaded blocks including a backing store indirect block having a plurality of fields, each backing store indirect block field corresponding to a field of the writable vdisk indirect block, one or more backing store indirect block fields having a pointer to a data block;

searching each field of the writable vdisk indirect block for a hole; and

filling each hole in the writeable vdisk by replacing each invalid pointer with a pointer to the data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.

Federwisch discloses a system and method "for remote asynchronous replication or mirroring of changes in a source file system snapshot in a destination replica file system using a scan (via a scanner) of the blocks that make up two versions of a snapshot of the source file system, which identifies changed blocks in the respective snapshot files..." *See* Federwisch, paragraph [0014]. Further, "[w]here a destination replicated snapshot may be needed at the source to, for example, rebuild the source qtree snapshot, (in other words, the role of the source and destination snapshot are reversed) the use of generalized rollback requires that the inode map be properly related between source and destination." *See* Federwisch, paragraph [0128]. "However, the inode map residing on the destination does not efficiently index the information in a form convenient for use by the source. Rather, the source would need to hunt randomly through the order presented in the map to obtain appropriate values." *See* Federwisch, paragraph [0128]. Thus, Federwisch performs "a 'flip' of map entries." *See* Federwisch, paragraph [0129]. Specifically, "the destination and source negotiate to transfer the inode map file to the source from the destination." *See* Federwisch, paragraph [0129]. "Next the source (which after the negotiation becomes the new destination), creates an empty inode map file with one entry for each inode of the source qtree." *See* Federwisch, paragraph [0130]. "[T]he new destination looks up the Nth inode from the entries associated with the old destination in the stored inode map file (i.e. the map from the old destination/new source)." (Emphasis added) *See* Federwisch, paragraph [0131]. If an entry in the old destination do not exist, then a "zero entry is created in the new inode map file, **representing that the Nth inode of the new source (old destination) is not allocated**." (Emphasis added). *See* Federwisch, paragraph [0131]. However, if there is an entry in the old destination, "a new entry [is created] in the new inode map file. The new entry maps the new source (old destination) Nth inode to the proper new destination (old source) inode." (Emphasis added). *See* Federwisch, paragraph [0131].

Edwards discloses a system and method for "performing an on-line checking [of] a file system in which inodes and directories comprising the file system are checked when first accessed." *See* Edwards, paragraph [0013]. Specifically, "[t]o check an inode, the buffer trees associated with the inode are verified in accordance with procedure 800 shown in FIG. 8. This procedure works by traversing the various branches of the buffer tree and verifying certain key items. First, in step 805, the inode check verifies that all pointers in the buffer tree are valid. If a pointer is directed to an invalid block, the pointer is cleared." *See* Edwards, paragraph [0048]. An invalid pointer is "a pointer [that] points to an invalid inode or file data block." *See* Edwards, paragraph [0054]. Specifically, the checking procedure in Edwards "ensures that all buffer trees have valid pointers, that any given block does not have multiple points to it, and other file system coherency checks." *See* Edwards, paragraph [0014]; *See also* Edwards, paragraph [0049].

The Applicant respectfully submits that a combination of Federwisch and Edwards does not teach or suggest the Applicant's claimed " … ***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store***" and "***filling each hole in the writeable vdisk by replacing each invalid pointer with a pointer to the data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.***"

In the Applicant's claimed technique, the occurrence of a hole in a writable vdisk instructs the system to examine a corresponding virtual block number pointer in the backing store and **fills each hole in the writable vdisk by replacing each invalid pointer with a pointer to the data block** referenced by the corresponding backing store indirect block field. Thus, the writable vdisk is updated to references both the data which is unchanged since the writable disk was created and the data which has been changed since the writable vdisk was created.

The Applicant respectfully submits that Federwisch fails to address these aspects of the Applicant's claims. Federwisch describes performing an Inode Map Flip, where a destination, that is updating its Inode Map, looks to source Inode Map and a zero value at a "particular position" in the source Inode Map causes the destination to put a zero value in the "same position" in its Inode Map. Specifically, in describing the Inode Map Flip, Federwisch states:

> [0128] Where a destination replicated snapshot may be needed at the source to, for example, rebuild the source qtree snapshot, (in other words, the role of the source and destination snapshot are reversed) the use of generalized rollback requires that the inode map be properly related between source and destination. This is because the source inodes do not match the destination inodes in their respective trees.

> [0129] One way to provide a source-centric inode map is to perform a "flip" of map entries. FIG. 17 details a procedure 1700 for performing the flip. (Emphasis added).
>
> [0130] Next the source (which after the negotiation becomes the new destination), creates an empty inode map file with one entry for each inode in the source qtree (step 1715). The new destination then initializes a counter with (in this example) N=1(step 1720). N is the variable representing the inode count on the new destination qtree.
>
> [0131] In step 1725, the new destination looks up the Nth inode from the entries associated with the old destination in the stored inode map file (i.e. the map from the old destination/new source). Next, the new destination determines if such an entry exists (decision step 1730). If no entry exists, then a zero entry is created in the new inode map file, representing that the Nth inode of the new source (old destination) is not allocated. However, if there exists an Nth inode of the new source/old destination, then the decision step 1730 branches to step 1740, and creates a new entry in the new inode map file (created in step 1715). The new entry maps the new source (old destination) Nth inode to the proper new destination (old source) inode. Note, in an alternate embodiment, the new inode map is provided with a full field of zero entries before the mapping begins, and the creation of a "zero entry," in this case should be taken broadly to include leaving a preexisting zero entry in place in the inode map.

As shown, the occurrence of a non-existent entry in the inode map at the source (e.g., the original destination) in Federwisch **causes the Federwisch system to place a zero in the new inode map entry at the destination (original source)**. In contrast, the occurrence of a "hole" in the Applicant's technique causes the Applicant's technique "***to examine a corresponding virtual block number pointer in the backing store***". The Applicant notes that placing a zero value that indicates that the inode is not in use as described in Federwisch, is very different than the Applicant's technique where a "hole" indicates that the a corresponding virtual block in the backing store should be examined. That is, the occurrence of a non-existent entry in Federwisch **does not** cause the Federwisch system to examine a corresponding vbn pointer in a different entity, but instead simply causes the Federwisch system to place a zero value in a destination Inode Map. As such, the Applicant respectfully submits that Federwisch may not fairly be interpreted to teach or suggest this aspect of the Applicant's claim.

Further, in the Applicant's technique, each hole of the writable vdisk is filled by replacing each invalid pointer with a pointer to a data block. Specifically, each hole in the writable vdisk is filled with the data block referenced by the corresponding backing store indirect block. Thus, the writeable vdisk in the Applicant's claims reference both the data which is unchanged since the writeable vdisk was created and the data which has been changed since the writeable vdisk was created. In contrast, and as explained above, a non-existent entry in Federwisch causes the Federwisch system to place a zero value in the new Inode Map. Thus, the Federwisch system is not "filling" holes with data, but is instead placing a zero value in its updated Inode Map when a zero value is encountered. Said differently, Federwisch makes no mention of encountering a non-existent entry and filling that non-existent entry with a reference to data. As such, the Applicant submits that Federwisch may not fairly be interpreted to teach or suggest the Applicant's claimed "***filling each hole in the writeable vdisk by replacing each invalid pointer with a pointer to the data block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created***."

Further, the Applicant respectfully submits that the deficiencies of Federwisch are not remedied by a combination with Edwards. Edwards states that if a pointer is directed to an invalid block, the pointer is cleared. *See* Edwards, paragraph [0048]. That is, the occurrence of an invalid pointer in Edwards causes the invalid point to be cleared, while the occurrence of a hole in the Applicant's claimed technique causes a corresponding virtual block in the backing store to be examined so that the hole can be filled with data referenced by the corresponding backing store indirect block.

Accordingly, the Applicant respectfully submits that a combination of Federwisch and Edwards is legally insufficient to render the present claims unpatentable under 35 U.S.C §103(a) because of the absence in Federwisch and Edwards of the Applicant's claimed " … ***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in the backing store***" and "***filling each hole in the writeable vdisk by replacing each invalid pointer with a pointer to the data***

block referenced by the corresponding backing store indirect block field to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created."

The present invention, as set forth in claim 40, representative of claims 40 – 43, comprises:

> 40. A method for operating a data storage system, comprising:
> creating a writable virtual disk (vdisk) at a selected time, the writable vdisk referencing changes in data stored in the data storage system after the writable vdisk was created, the writable vdisk having a plurality of holes ***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in a backing store***;
> maintaining the backing store, the backing store referencing the data stored in the data storage system which has not been changed since the writable vdisk was created;
> searching, by a background task process, each field of the writable vdisk for a hole;
> ***for each hole in the writeable vdisk, marking as dirty the corresponding data block pointed to by the backing store indirect block without modifying the corresponding data block***; and
> ***performing a write allocation to replace each hole in the writable vdisk to point to the data block marked as dirty and referenced by the corresponding backing store indirect block to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created.***

Independent claims 40 – 43 have been amended to include a limitation similar to that of claim 2. The Applicant notes that claim 2 was rejected under 35 U.S.C §103(a) as being unpatentable over Federwisch in view of Edwards, in further view of Haskin. As such, the Applicant will address the rejection of claims 40 – 43 with respect to Federwisch, Edwards, and Haskin.

Federwisch and Edwards are described above.

Haskin describes a technique for storing snapshots that indicate a status of stored data at a particular point in time. *See* Haskin, paragraph [0002]. In particular, Haskin describes moving "old" data to a snapshot dataset before having "new" data be referenced by indirect blocks that originally referenced the "old" data. *See* Haskin, paragraph [0079].

The Applicant respectfully submits that a combination of Federwisch, Edwards, and Haskin does not teach or suggest the Applicant's claimed "***where each hole instructs the data storage system to examine a corresponding virtual block number pointer in a backing store***" and "***performing a write allocation to replace each hole in the writable vdisk to point to the data block marked as dirty and referenced by the corresponding backing store indirect block to update the writable vdisk to reference both the data which is unchanged since the writable vdisk was created and the data which has been changed since the writable vdisk was created***."

First, for at least the reasons asserted above, Federwisch and Edwards fails to address these aspects of the Applicant's claims. Further, deficiencies of Federwisch and Edwards are not remedied by a combination with Haskin. Instead, Haskin simply describes a technique for storing snapshots.

Moreover, a combination of Federwisch, Edwards, and Haskin does not teach or suggest the Applicant's claimed "***for each hole in the writeable vdisk, marking as dirty the corresponding data block pointed to by the backing store indirect block without modifying the corresponding data block***."

There appears to be agreement that Federwisch and Edwards do not address these aspects of the Applicant's claims. *See* Office Action, page 17.

The deficiencies of Federwisch and Edwards, however, are not remedied by a combination with Haskin. Instead, Haskin simply describes moving "old" data to a snapshot dataset before having "new" data be referenced by indirect blocks that originally referenced the "old" data. Specifically, Haskin states:

> [0079] If an operation to the original data block <u>overwrites a whole data block</u>, some embodiments of the present invention do not copy the data to a new disk block prior to being updated. These <u>embodiments move the old disk block into the snapshot dataset</u> by storing the disk address of the original data block into the indirect block stored in the snapshot dataset <u>and replacing the address in the indirect block of the original file with the address of a newly allocated data block</u>. This saves the disk I/O associated with copying the original data from the original data block into the

> snapshot dataset when the original file will no longer need the original data. (Emphasis added).

Thus, Haskin is in fact altering the data content of the data block because the "new" data takes the place of the "old" data. As such, Haskin may not fairly be interpreted to teach or suggest the Applicant's technique that "dirties" the data block pointed to by the backing store indirect block **without altering a data content of the data block**.

Accordingly, the Applicant respectfully submits that a combination of Federwisch, Edwards, and Haskin is legally insufficient to render the present claims unpatentable under 35 U.S.C §103(a).

At pages 17 – 20 of the Office Action claims 2, 4, 5, 11, 12, 24, 26, 27, 32, 34, and 35 were rejected under 35 U.S.C §103(a) as being unpatentable over Federwisch in view of Edwards, in further view of Haskin.

The Applicant notes that claims 2, 4, 5, 11, 12, 24, 26, 27, 32, 34, and 35 are dependent claims that depend from independent claims believed to be in condition for allowance. Accordingly, claims 2, 4, 5, 11, 12, 24, 26, 27, 32, 34, and 35 are believed to be in condition for allowance due to their dependency as well as for other separate reasons.

Conclusion

In summary, all the independent claims are believed to be in condition for allowance and therefore all dependent claims that depend there from are believed to be in condition for allowance. The Applicant respectfully solicits favorable action.

Please charge any additional fee occasioned by this paper to our Deposit Account No. 03-1237.

Respectfully submitted,

/Omar M. Wadhwa/

Omar M. Wadhwa
Reg. No. 64,127
CESARI AND MCKENNA, LLP
88 Black Falcon Avenue
Boston, MA 02210-2414
(617) 951-2500